Journal of Scientific Computing (2026) 106:8
https://doi.org/10.1007/s10915-025-03144-0

Numerical Aspects of the Tensor Product Multilevel Method for High-Dimensional, Kernel-Based Reconstruction on Sparse Grids

Markus Büttner[1] · **Rüdiger Kempf**[2] · **Holger Wendland**[2]

Received: 21 January 2025 / Revised: 25 September 2025 / Accepted: 17 November 2025 /
Published online: 2 December 2025

Abstract
This paper investigates the approximation of functions with finite smoothness defined on domains with a Cartesian product structure. The recently proposed *tensor product multilevel method (TPML)* combines Smolyak's sparse grid method with a kernel-based residual correction technique. The contributions of this paper are twofold. First, we present two improvements on the TPML that reduce the computational cost of point evaluations compared to a naive implementation. Second, we provide numerical examples that demonstrate the effectiveness and innovation of the TPML.

Keywords Radial basis functions · sparse grids · multilevel approximation

Mathematics Subject Classification 65D12 · 65D15 · 65D40

1 Introduction

Many current problems in physics, engineering, and even business science are higher-dimensional in nature. This could be because the problem is modeled as a partial differential equation in space-time, which results in a two/three-plus-one-dimensional problem, or because it involves uncertainty, which is modeled by a stochastic process and then parameterized, done so in uncertainty quantification. The number of parameters then corresponds to the dimension of the problem. Classical numerical methods are often numerically not feasible, even in only four-dimensional examples, because of the curse of dimensionality. One way to alleviate this curse is by means of Smolyak's method [1]. This method is tailored for problems whose domain can be decomposed into the Cartesian product of lower-dimensional

✉ Rüdiger Kempf
ruediger.kempf@uni-bayreuth.de

Markus Büttner
markus.buettner@uni-bayreuth.de

Holger Wendland
holger.wendland@uni-bayreuth.de

[1] Chair of Scientific Computing, University of Bayreuth, Bayreuth, Germany

[2] Applied and Numerical Analysis, University of Bayreuth, Bayreuth, Germany

domains, for example space and time, and allows us to combine numerical methods used on these lower-dimensional domains to an approximation process in high dimensions and offers a very flexible ansatz with applications in quadrature [1–4], interpolation [5–7], solving PDEs [8–10], machine learning [11] and even financial mathematics [12]. The list given here is by no means exhaustive. For a more complete list, we refer to [13, 14].

Recently, a new method, called the *tensor product multilevel method (TPML)*, was proposed and studied in [7], combining Smolyak's method with a kernel-based multilevel scheme [15–18]. This method allows us to approximate moderately high-dimensional functions defined on the Cartesian product of lower-dimensional domains using scattered data. The ability to use general lower-dimensional domains instead of only intervals is one of the exceptional features of this method. Other known methods are either spline or polynomial based and are therefore based on intervals and thus fail at solving these kinds of problems. There have been previous attempts to combine Smolyak's method with a kernel-based approach, see, e.g., [19], but concentrated on non-compactly supported basis functions and did not provide a rigorous error analysis. In contrast, the paper [7] provides a thorough error analysis for compactly supported kernels of finite smoothness. There, an explicit representation of the approximation operator is also given. However, it turns out that a naive implementation of the derived representation is not feasible because the computational cost is too large. Hence, in the present paper, we propose two different reformulations of the TPML, with the result that the method can actually be applied to real-world examples. The first reformulation strives to relocate expensive computations into an offline phase such that the numerical cost of several point evaluations of the approximation becomes cheaper compared to the naive implementation. The second reformulation is tailored to problems with nested sets of sites, which occur naturally in typical applications. There, we use the nestedness to derive a representation of the TPML that uses every data value only once, reducing the numerical costs this way.

The outline of this paper is as follows. In Section 2 we briefly recall the building blocks of the TPML, i.e., Smolyak's method and the formulation of the kernel-based multilevel method using a (modified) Lagrange basis, before giving the definition of the TPML operator. In Section 3 we introduce the two variations of the formulation of TPML. And finally in Section 4 we demonstrate the power of the new formulations by applying the TPML to a real-world application: the tidal flow at the Bight of Abaco, an example of an interpolation of simulation data of a space-time differential equation, and to a benchmarking problem: a cantilever beam with seven parameters. To keep Section 3 brief, we give the proofs of the results in Appendices A and B.

2 The Tensor Product Multilevel Method

We briefly recall the basic ideas of the kernel-based tensor product multilevel method; for details we refer to [7]. The approach combines the kernel-based multilevel method for scattered data approximation on low-dimensional domains with Smolyaks construction. This way, one obtains a high-dimensional approximation scheme that essentially inherits the stability and approximation properties of the underlying low-dimensional building blocks.

2.1 Smolyak's Method

Smolyak's construction and subsequent generalizations, as well as the numerical approximation power and computational effort depend on a pre-determined *index set*. This set describes

how many levels we will use in each direction. In order to have the maximum freedom and to adapt to possible different smoothnesses in different directions it became customary to use *anisotropic index sets*, see, e.g., [4, 5, 9, 20].

Definition 2.1 For $d \in \mathbb{N}$ let $\boldsymbol{\omega} \in \mathbb{R}_+^d$ and $\ell \in \mathbb{N}$. The *anisotropic index set* $\mathcal{I}_{\boldsymbol{\omega}}(\ell, d) \subseteq \mathbb{N}^d$ is defined by

$$\mathcal{I}_{\boldsymbol{\omega}}(\ell, d) := \left\{ \boldsymbol{\lambda} \in \mathbb{N}^d \ : \ \sum_{j=1}^{d} (\lambda_j - 1)\omega_j \leq \ell \omega_{min} \right\}, \tag{1}$$

where $\omega_{min} := \min_{1 \leq j \leq d} \omega_j$.

Its *surface* $\mathcal{J}_{\boldsymbol{\omega}}(\ell, d)$ is defined by

$$\mathcal{J}_{\boldsymbol{\omega}}(\ell, d) = \mathcal{I}_{\boldsymbol{\omega}}(\ell, d) \setminus \mathcal{I}_{\boldsymbol{\omega}} \left(\ell - \frac{\|\boldsymbol{\omega}\|_1}{\omega_{min}}, d \right). \tag{2}$$

The positive weight vector $\boldsymbol{\omega}$ reflects how important each direction is. The larger the quotient ω_j / ω_{min} is chosen the less important direction j is for us and $\mathcal{I}_{\boldsymbol{\omega}}(\ell, d)$ is less extended in this direction. For fixed $\boldsymbol{\omega}$ the threshold ℓ determines the maximum number of levels in each direction. The largest level in each direction can be directly computed by

$$\lambda_{j,max} = \left\lfloor \frac{\ell \omega_{min}}{\omega_j} \right\rfloor + 1.$$

Remark 2.2 For the error analysis in [7], we have to assume that the weight vector $\boldsymbol{\omega}$ is ascendingly ordered, i.e., $\omega_1 \leq \omega_2 \leq \cdots \leq \omega_d$.

Given sequences of operators $A_i^{(j)}$, $1 \leq j \leq d$, $1 \leq i \leq \lambda_{j,max}$, and setting $A_0^{(j)} = 0$, the Smolyak operator in combination technique representation is given as

$$\mathcal{S}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)} := \sum_{\boldsymbol{\lambda} \in \mathcal{J}_{\boldsymbol{\omega}}(\ell,d)} \sum_{\substack{\boldsymbol{\beta} \in \{0,1\}^d \\ \boldsymbol{\lambda} + \boldsymbol{\beta} \in \mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} (-1)^{\|\boldsymbol{\beta}\|_1} \left(A_{\lambda_1}^{(1)} \otimes \cdots \otimes A_{\lambda_d}^{(d)} \right).$$

For a thorough study of the tensor product of linear spaces and operators we refer to [21].

Smolyak's method is deeply connected to *sparse grids*, which we define now. To families of sites $X_{\lambda_j}^{(j)}$, $1 \leq \lambda_j \leq \lambda_{j,max}$, the sparse grid $\mathcal{H}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)} \subseteq \Omega$ is defined as

$$\mathcal{H}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)} := \bigcup_{\boldsymbol{\lambda} \in \mathcal{I}_{\boldsymbol{\omega}}(\ell,d)} X_{\lambda_1}^{(1)} \otimes \cdots \otimes X_{\lambda_d}^{(d)}.$$

2.2 Kernel Multilevel Method using Lagrange Functions

The standard kernel-based multilevel method, described in, e.g., [15], is a residual correction scheme for scattered data approximation. It can be used to beat the *trade-off principle* in scattered data approximation, essentially stating that one can either have stable computation of the solution but no convergence of the approximation error, or convergence but unstable, i.e., badly conditioned system matrices. By using the multilevel method one has both, well conditioned matrices and convergence.

The key ingredients are a sequence of, not necessarily nested, sets of sites $X_1, \ldots, X_L$, level-dependently scaled kernels Φ_i and associated *local approximation spaces*

$$W_i = \operatorname{span} \left\{ \Phi_i(\cdot - \boldsymbol{x}_{i,k}) \ : \ \boldsymbol{x}_{i,k} \in X_i \right\}, \quad 1 \leq i \leq L.$$

In the literature [15–18], one usually fixes one radial basis function $\Phi : \mathbb{R}^n \to \mathbb{R}$ with compact support and scales it with a level-dependent scaling parameter $\delta_i > 0$ to obtain $\Phi_i := \delta_i^{-n} \Phi(\cdot/\delta_i)$.

The multilevel method then determines for every $1 \le i \le L$ the local approximation $s_i \in W_i$ to the residual e_{i-1} of the previous level, either by interpolation or penalized least squares approximation, and updates the global approximation $f_i \in W_1 + \cdots + W_i$ and the residual e_i according to

$$\begin{aligned} f_i &= f_{i-1} + s_i, \\ e_i &= e_{i-1} - s_i. \end{aligned}$$

In this standard formulation of the kernel-based multilevel method, there is only an implicit dependence of the data. Hence, the authors of [7] found a new representation of the method in terms of *Lagrange* or *cardinal functions* $\{\chi_{i,k}\}$, that made the dependence on the data explicit. The defining property of Lagrange functions is that they satisfy $\chi_{i,k}(\boldsymbol{x}_{i,m}) = \delta_{km}$ for $\boldsymbol{x}_{i,m} \in X_i$.

The Lagrange basis can be computed by solving a linear system. However, the matrix differs whether we use interpolation or penalized least squares approximation. We remark that in the case of penalized least squares, the functions do not satisfy the Lagrange condition, but lead to good numerical results anyway.

Definition 2.3 For $L \in \mathbb{N}$, $1 \le i \le L$ and $1 \le k \le N_i$ let

$$\boldsymbol{r}_i := \big(\Phi_i(\cdot - \boldsymbol{x}_{i,1}), \ldots, \Phi_i(\cdot - \boldsymbol{x}_{i,N_i})\big)^{\mathrm{T}} \in \mathbb{R}^{N_i}. \tag{3}$$

The k-th Lagrange function $\chi_{i,k}$ on level i can then be expressed as

$$\chi_{i,k} = \big(\boldsymbol{\alpha}_{i,k}\big)^{\mathrm{T}} \boldsymbol{r}_i. \tag{4}$$

The coefficient vector $\boldsymbol{\alpha}_i \in \mathbb{R}^{N_i}$ is given as

$$\boldsymbol{\alpha}_{i,k} := \begin{cases} M_i^{-1} \boldsymbol{e}_k, & \text{for interpolation,} \\ (M_i + \kappa_i I)^{-1} \boldsymbol{e}_k, & \text{for penalized least-squares approximation} \end{cases} \tag{5}$$

where $(M_i)_{k,m} := \big(\Phi_i(\boldsymbol{x}_{i,k} - \boldsymbol{x}_{i,m})\big)$ is the Gramian, $\boldsymbol{e}_k$ denotes the k-th unit vector and $I \in \mathbb{R}^{N_i \times N_i}$ is the unit matrix and κ_i are chooseable regularization parameter.

We use the following notation throughout the rest of this paper.

Definition 2.4 1. For $L \in \mathbb{N}$ we call a subset $\mathfrak{u} := \{u_1, \ldots, u_m\} \subseteq \{1, \ldots, L\}$ with $m = \#\mathfrak{u}$ elements *ordered*, if $u_1 < u_2 < \cdots < u_m$.
2. For an ordered set $\mathfrak{u} = \{u_1, \ldots, u_m\}$, we take $\boldsymbol{N}_\mathfrak{u}$ as $\boldsymbol{N}_\mathfrak{u} := (N_{u_1}, \ldots, N_{u_m})^{\mathrm{T}} \in \mathbb{N}^{\#\mathfrak{u}}$.
3. For an ordered set $\mathfrak{u} = \{u_1, \ldots, u_m\}$ we abbreviate

$$\sum_{\boldsymbol{k} \le \boldsymbol{N}_\mathfrak{u}} := \sum_{k_1 \le N_{u_1}} \cdots \sum_{k_m \le N_{u_m}},$$

where we implicitly assume that the dimension of $\boldsymbol{k}$ fits the dimension of $\boldsymbol{N}_\mathfrak{u}$.

Then, the multilevel approximation operator can be expressed in the following way, see [7, Theorem 3.9].

Theorem 2.5 *Let* $\mathfrak{u} = \{u_1, \ldots, u_{\#\mathfrak{u}}\}$ *be an ordered set. Define the* combined operator $\mathcal{I}_\mathfrak{u} : C(\Omega) \to W_{\#\mathfrak{u}}$ *by*

$$\mathcal{I}_\mathfrak{u} f := \sum_{\boldsymbol{k} \le \boldsymbol{N}_\mathfrak{u}} a(\mathfrak{u}, \boldsymbol{k}) f(\boldsymbol{x}_{u_1,k_1}) \chi_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}, \tag{6}$$

where the coefficients are given by $a(\mathfrak{u}, \boldsymbol{k}) = 1$, *if* $\#\mathfrak{u} = 1$ *and*

$$a(\mathfrak{u}, \boldsymbol{k}) = \prod_{\ell=1}^{\#\mathfrak{u}-1} \chi_{u_\ell,k_\ell}(\boldsymbol{x}_{u_{\ell+1},k_{\ell+1}})$$

for $\#\mathfrak{u} > 1$.

Then the multilevel approximation operator $A_L : C(\Omega) \to \bigoplus_{i=1}^{L} W_i$ *at level* L *has the representation*

$$A_L(f) = \sum_{\substack{\mathfrak{u} \subseteq \{1,\ldots,L\} \\ 1 \le \#\mathfrak{u} \le L}} (-1)^{\#\mathfrak{u}+1} \sum_{\boldsymbol{k} \le \boldsymbol{N}_\mathfrak{u}} a(\mathfrak{u}, \boldsymbol{k}) f(\boldsymbol{x}_{u_1,k_1}) \chi_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}$$

for $f \in C(\Omega)$.

2.3 The Tensor Product Multilevel Method

The combination of Smolyak's method and the kernel-based multilevel method yields the tensor product multilevel approximation operator for continuous functions on $\Omega^\otimes = \Omega^{(1)} \times \cdots \times \Omega^{(d)}$, where $\Omega^{(j)} \subseteq \mathbb{R}^{n_j}$, $1 \le j \le d$ and $n_j \in \mathbb{N}$. In particular, we allow $n_j > 1$.

Definition 2.6 We define the *tensor product multilevel approximation* to $\boldsymbol{f} \in C(\Omega^\otimes)$, evaluated at $\boldsymbol{x} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(d)}) \in \Omega^\otimes$, as

$$\begin{aligned}
&\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(\boldsymbol{f})(\boldsymbol{x}) := \\
&= \sum_{\boldsymbol{\lambda} \in \mathcal{J}_{\boldsymbol{\omega}}(\ell,d)} \sum_{\substack{\boldsymbol{\beta} \in \{0,1\}^d \\ \boldsymbol{\lambda} + \boldsymbol{\beta} \in \mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} \sum_{\substack{\mathfrak{u}^{(1)} \subseteq \{1,\ldots,\lambda_1\} \\ 1 \le \#\mathfrak{u}^{(1)} \le \lambda_1}} \cdots \sum_{\substack{\mathfrak{u}^{(d)} \subseteq \{1,\ldots,\lambda_d\} \\ 1 \le \#\mathfrak{u}^{(d)} \le \lambda_d}} c_{\boldsymbol{\beta}}(\mathfrak{u}^{(1)}, \ldots, \mathfrak{u}^{(d)}) \cdot \\
&\qquad \cdot \left(\mathcal{I}^{(1)}_{\mathfrak{u}^{(1)}} \otimes \cdots \otimes \mathcal{I}^{(d)}_{\mathfrak{u}^{(d)}} \right) (\boldsymbol{f})(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{\lambda} \in \mathcal{J}_{\boldsymbol{\omega}}(\ell,d)} \sum_{\substack{\boldsymbol{\beta} \in \{0,1\}^d \\ \boldsymbol{\lambda} + \boldsymbol{\beta} \in \mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} \sum_{\substack{\mathfrak{u}^{(1)} \subseteq \{1,\ldots,\lambda_1\} \\ 1 \le \#\mathfrak{u}^{(1)} \le \lambda_1}} \cdots \sum_{\substack{\mathfrak{u}^{(d)} \subseteq \{1,\ldots,\lambda_d\} \\ 1 \le \#\mathfrak{u}^{(d)} \le \lambda_d}} c_{\boldsymbol{\beta}}(\mathfrak{u}^{(1)}, \ldots, \mathfrak{u}^{(d)}) \\
&\qquad \sum_{\boldsymbol{k}^{(1)} \le \boldsymbol{N}_{\mathfrak{u}^{(1)}}} \cdots \sum_{\boldsymbol{k}^{(d)} \le \boldsymbol{N}_{\mathfrak{u}^{(d)}}} \boldsymbol{f}(\boldsymbol{x}^{(1)}_{u^{(1)}_1,k^{(1)}_1}, \ldots, \boldsymbol{x}^{(d)}_{u^{(d)}_1,k^{(d)}_1}) \cdot \\
&\qquad \cdot \prod_{j=1}^{d} a^{(j)}(\mathfrak{u}^{(j)}, \boldsymbol{k}^{(j)}) \cdot \left(\prod_{j=1}^{d} \chi^{(j)}_{u^{(j)}_{\#\mathfrak{u}^{(j)}},k^{(j)}_{\#\mathfrak{u}^{(j)}}} (\boldsymbol{x}^{(j)}) \right),
\end{aligned} \tag{7}$$

where

$$c_{\boldsymbol{\beta}}(\mathfrak{u}^{(1)}, \ldots, \mathfrak{u}^{(d)}) = (-1)^{\|\boldsymbol{\beta}\|_1 + d + \#\mathfrak{u}^{(1)} + \cdots + \#\mathfrak{u}^{(d)}}$$

and $a^{(j)}(\mathfrak{u}^{(j)}, \boldsymbol{k}^{(j)}) = 1$ if $\#\mathfrak{u}^{(j)} = 1$ and

$$a^{(j)}(\mathfrak{u}^{(j)}, \boldsymbol{k}^{(j)}) = \prod_{m=1}^{\#\mathfrak{u}^{(j)}-1} \chi_{u_m^{(j)},k_m^{(j)}}(\boldsymbol{x}^{(j)}_{u_{m-1}^{(j)},k_{m-1}^{(j)}}) \tag{8}$$

for $\#\mathfrak{u}^{(j)} > 1$ for every $1 \le j \le d$.

Remark 2.7 1. We can see that the target function is only evaluated in the sites $\boldsymbol{x}^{(j)}_{u_1^{(j)},k_1^{(j)}}$, $1 \le k_1^{(j)} \le N^{(j)}_{u_1^{(j)}}$. These are the points in the coarsest set of sites $X^{(j)}_{u_1^{(j)}}$ associated to each ordered set $\mathfrak{u}$.
2. Similarly, we see that only the Lagrange functions associated to the finest set of sites for each ordered set $\mathfrak{u}$ are evaluated at the point $\boldsymbol{x}$.

Although this is a valid representation of the approximation operator, a naive implementation is, even in only two directions and a moderate number of levels in each direction, numerically too expensive. This can heuristically be seen by simply counting the number of subsets $\mathfrak{u}^{(j)} \subseteq \{1, \dots, \lambda_j\}$ and the consequently needed combinations of all points on each associated level in (7). See also the discussion in Section 4.3.

3 Modifications of the Tensor Product Multilevel Method

The aim is now to reduce the online cost of a point-evaluation of the tensor product multilevel approximation. We present two ways to achieve this goal. First, by moving expensive computations to an offline phase. And second, by deriving a completely new representation of the approximation operator, where we change the perspective and sum of the points in the associated sparse grid rather than the multi-indices.

3.1 Towards more efficient point-evaluations

To keep this section brief we move the proofs into Appendix A. Most of the ideas are technical transformations involving the elements of the ordered sets $\mathfrak{u}^{(j)}$.

The approximation $\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(\boldsymbol{f})$ in (7) is essentially a linear combination of the tensor product of the combined operators $\mathcal{I}_{\mathfrak{u}}$ given in (6). Hence, the first step is to simplify the representation of $\mathcal{I}^{(1)}_{\mathfrak{u}^{(1)}} \otimes \cdots \otimes \mathcal{I}^{(d)}_{\mathfrak{u}^{(d)}}$. To do this, we take a closer look at each of the direction-wise, combined operator $\mathcal{I}^{(j)}_{\mathfrak{u}^{(j)}}$ and try to separate the intermediate terms in $a^{(j)}(\mathfrak{u}^{(j)}, \boldsymbol{k}^{(j)})$. These can be pre-computed since they are independent of the target function and the evaluation point. This idea can be applied in each direction, hence we omit the superscript (j) in the next lemma.

Lemma 3.1 *Let $\mathfrak{u} \subseteq \{1, \dots, L\}$ be a fixed ordered set and*

$$\boldsymbol{\alpha}_{i,k} := \begin{cases} M_i^{-1}\boldsymbol{e}_k, & \text{for interpolation,} \\ (M_i + \kappa_i I)^{-1}\boldsymbol{e}_k, & \text{for penalized least-squares approximation} \end{cases}$$

be the coefficient vector for $\chi_{i,k}$.

Define $P_{\mathfrak{u}} \in \mathbb{R}^{N_{u_1} \times N_{u_{\#\mathfrak{u}}}}$ *as*

$$P_{\mathfrak{u}} := \prod_{m=2}^{\#\mathfrak{u}} \sum_{k_m \le N_{u_m}} \boldsymbol{r}_{u_{m-1}}(\boldsymbol{x}_{u_m,k_m}) \boldsymbol{\alpha}^T_{u_m,k_m}, \tag{9}$$

with $\boldsymbol{r}_i$ *as in* (3).

Then we can express the combined local operators $\mathcal{I}_{\mathfrak{u}}$ *as*

$$\mathcal{I}_{\mathfrak{u}} f(\boldsymbol{x}) = \sum_{k_1 \le N_{u_1}} f(\boldsymbol{x}_{u_1,k_1}) \boldsymbol{\alpha}^T_{u_1,k_1} \cdot P_{\mathfrak{u}} \cdot \boldsymbol{r}_{u_{\#\mathfrak{u}}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega.$$

Remark 3.2 1. To compute the vectors $(\boldsymbol{\alpha}_{i,k})_{1 \le k \le N_i} \in \mathbb{R}^{N_i}$ we have to solve N_i-many sparse linear systems, where the matrix stays the same only the right-hand side changes.
2. We recall that $\boldsymbol{r}_i(\boldsymbol{x})$ is the evaluation vector of the rescaled kernel on level i. It has only constant many non-zero entries. This means that each of the rank-1-matrices

$$\boldsymbol{r}_{u_{m-1}}(\boldsymbol{x}_{u_m,k_m}) \boldsymbol{\alpha}^{\mathrm{T}}_{u_m,k_m}$$

in (9) is sparse. However, we cannot leverage this since for every k_m the sparsity pattern differs.
3. All matrices $P_{\mathfrak{u}}$ can be computed in an offline phase and are independent of the evaluation point $\boldsymbol{x}$ and the data $f(\boldsymbol{x}_{u_1,k_1})$. Additionally, all these computations are vector or matrix computations and, therefore, can be done efficiently on GPUs.

We can now use Lemma 3.1 for every direction $1 \le j \le d$ and obtain a new representation of the tensor product combined operator.

Theorem 3.3 *Use the notation and assumptions of Definition 2.6. Additionally, assume that for every* $1 \le j \le d$ *and every ordered set* $\mathfrak{u}^{(j)}$ *we use the notation of Lemma 3.1. Set*

$$\boldsymbol{x}_{\boldsymbol{u}_1,\boldsymbol{k}_1} := (\boldsymbol{x}^{(1)}_{u_1^{(1)},k_1^{(1)}}, \dots, \boldsymbol{x}^{(d)}_{u_1^{(d)},k_1^{(d)}}).$$

Then the tensor product combined operator can be expressed as

$$\begin{aligned} \left(\mathcal{I}^{(1)}_{\mathfrak{u}^{(1)}} \otimes \cdots \otimes \mathcal{I}^{(d)}_{\mathfrak{u}^{(d)}}\right)(\boldsymbol{f})(\boldsymbol{x}) = \\ = \sum_{k_1^{(1)} \le N^{(1)}_{u_1^{(1)}}} \cdots \sum_{k_1^{(d)} \le N^{(d)}_{u_1^{(d)}}} f(\boldsymbol{x}_{\boldsymbol{u}_1,\boldsymbol{k}_1}) \cdot \\ \cdot \prod_{j=1}^{d} (\boldsymbol{\alpha}^{(j)}_{u_1^{(j)},k_1^{(j)}})^T \cdot P^{(j)}_{\mathfrak{u}^{(j)}} \cdot \boldsymbol{r}^{(j)}_{u^{(j)}_{\#\mathfrak{u}(j)}}(\boldsymbol{x}^{(j)}). \end{aligned} \tag{10}$$

The new representation of $\left(\mathcal{I}^{(1)}_{\mathfrak{u}^{(1)}} \otimes \cdots \otimes \mathcal{I}^{(d)}_{\mathfrak{u}^{(d)}}\right)(\boldsymbol{f})(\boldsymbol{x})$ immediately yields a simplified intermediate representation of the multilevel tensor product approximation.

Corollary 3.4 *With the notation and assumptions of Theorem 3.3 the tensor product multilevel approximant $\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(\boldsymbol{f})$ can, for every $\boldsymbol{x} \in \Omega$, be expressed as*

$$\begin{aligned}\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(\boldsymbol{f})(\boldsymbol{x}) = &\sum_{\boldsymbol{\lambda}\in\mathcal{J}_{\boldsymbol{\omega}}(\ell,d)} \sum_{\substack{\boldsymbol{\beta}\in\{0,1\}^d \\ \boldsymbol{\lambda}+\boldsymbol{\beta}\in\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} (-1)^{\|\boldsymbol{\beta}\|_1+d} \\ &\sum_{u_1^{(1)}\in\{1,\dots,\lambda_1\}} \cdots \sum_{u_1^{(d)}\in\{1,\dots,\lambda_d\}} \sum_{k_1^{(1)}\le N^{(1)}_{u_1^{(1)}}} \cdots \sum_{k_1^{(d)}\le N^{(d)}_{u_1^{(d)}}} \boldsymbol{f}(\boldsymbol{x}_{u_1,k_1})\cdot \\ &\prod_{j=1}^{d}\left(\boldsymbol{\alpha}^{(j)}_{u_1^{(j)},k_1^{(j)}}\right)^T \sum_{\substack{\widetilde{\mathfrak{u}}^{(j)}\subseteq\{1,\dots,\lambda_j\} \\ \widetilde{u}_1^{(j)}=u_1^{(j)}}} (-1)^{\#\widetilde{\mathfrak{u}}^{(j)}} P^{(j)}_{\widetilde{\mathfrak{u}}^{(j)}} \cdot \boldsymbol{r}^{(j)}_{\widetilde{u}^{(j)}_{\#\widetilde{\mathfrak{u}}^{(j)}}}(\boldsymbol{x}^{(j)}).\end{aligned} \tag{11}$$

Remark 3.5 The ordered sets $\widetilde{\mathfrak{u}}^{(j)}$ in the inner most sum in (11) are those ordered sets $\mathfrak{u}^{(j)}$ of (7) whose first element is the, for the inner sum fixed, $u_1^{(j)}$.

The final step is now to simplify the inner most sum in (11). This will be achieved by also fixing the last element of the ordered sets $\widetilde{\mathfrak{u}}$. This allows us to pre-compute most of the intermediate sums over the matrices $P^{(j)}_{\widetilde{\mathfrak{u}}^{(j)}}$.

Corollary 3.6 *For $1 \le m \le p \le \lambda_j$, $1 \le j \le d$, set*

$$S^{(j)}_{m,p} := \sum_{\substack{\overline{\mathfrak{u}}^{(j)}\subseteq\{1,\dots,\lambda_j\} \\ \overline{u}_1^{(j)}=m \\ \overline{u}_d^{(j)}=p}} (-1)^{\#\overline{\mathfrak{u}}^{(j)}} P^{(j)}_{\overline{\mathfrak{u}}^{(j)}} \quad \in \mathbb{R}^{N_m^{(j)}\times N_p^{(j)}}. \tag{12}$$

Then we have

$$\sum_{\substack{\widetilde{\mathfrak{u}}^{(j)}\subseteq\{1,\dots,\lambda_j\} \\ \widetilde{u}_1^{(j)}=u_1^{(j)}}} (-1)^{\#\widetilde{\mathfrak{u}}^{(j)}} P^{(j)}_{\widetilde{\mathfrak{u}}^{(j)}} \cdot \boldsymbol{r}^{(j)}_{\widetilde{u}^{(j)}_{\#\widetilde{\mathfrak{u}}^{(j)}}}(\boldsymbol{x}^{(j)}) = \sum_{p=u_1^{(j)}}^{\lambda_j} S^{(j)}_{u_1^{(j)},p}\cdot \boldsymbol{r}^{(j)}_p(\boldsymbol{x}^{(j)}). \tag{13}$$

Remark 3.7 1. The matrix $S^{(j)}_{m,p}$ in (12) considers all matrices $P_{\mathfrak{u}}$ whose ordered set $\overline{\mathfrak{u}}^{(j)}$ has m as first and p as last entry.
2. We can express the sum on the right-hand side of (13) as

$$\begin{pmatrix} S^{(j)}_{u_1^{(j)},u_1^{(j)}} & S^{(j)}_{u_1^{(j)},u_1^{(j)}+1} & \cdots & S^{(j)}_{u_1^{(j)},\lambda_j}\end{pmatrix} \begin{pmatrix} \boldsymbol{r}^{(j)}_{u_1^{(j)}}(\boldsymbol{x}^{(j)}) \\ \boldsymbol{r}^{(j)}_{u_1^{(j)}+1}(\boldsymbol{x}^{(j)}) \\ \vdots \\ \boldsymbol{r}^{(j)}_{u_{\lambda_j}^{(j)}}(\boldsymbol{x}^{(j)})\end{pmatrix}.$$

3. We collect the matrices $S^{(j)}_{m,p}$ for all $1 \le m \le p \le \lambda_{j,max}$ in the system

$$\boldsymbol{S}^{(j)} := \begin{pmatrix} S^{(j)}_{1,1} & S^{(j)}_{1,2} & \cdots & S^{(j)}_{1,\lambda_{j,max}} \\ & S^{(j)}_{2,2} & \cdots & S^{(j)}_{2,\lambda_{j,max}} \\ & & \ddots & \vdots \\ & & & S^{(j)}_{\lambda_{j,max},\lambda_{j,max}} \end{pmatrix}. \tag{14}$$

Finally, we can interpret the term in the product in (11),

$$\left(\boldsymbol{\alpha}^{(j)}_{u^{(j)}_1,k^{(j)}_1}\right)^{\mathrm{T}} \sum_{p=u^{(j)}_1}^{\lambda_j} S^{(j)}_{u^{(j)}_1,p} \cdot \boldsymbol{r}^{(j)}_p(\boldsymbol{x}^{(j)})$$

as a product of block-matrices. Using the notation

$$\boldsymbol{A}^{(j)}_i = \begin{pmatrix} \boldsymbol{\alpha}^{(j)}_{i,1} \\ \vdots \\ \boldsymbol{\alpha}^{(j)}_{i,N^{(j)}_i} \end{pmatrix},$$

we define the matrix $\boldsymbol{\Xi}^{(j)}(\boldsymbol{x}^{(j)})$ by

$$\begin{aligned} \boldsymbol{\Xi}^{(j)}(\boldsymbol{x}^{(j)}) &:= \begin{pmatrix} \boldsymbol{\xi}^{(j)}_{1,1}(\boldsymbol{x}^{(j)}) & \cdots & \boldsymbol{\xi}^{(j)}_{1,\lambda_{j,max}}(\boldsymbol{x}^{(j)} \\ & \ddots & \vdots \\ 0 & & \boldsymbol{\xi}^{(j)}_{\lambda_{j,max},\lambda_{j,max}}(\boldsymbol{x}^{(j)}) \end{pmatrix} \\ &= \begin{pmatrix} A^{(j)}_i & & 0 \\ & \ddots & \\ 0 & & A^{(j)}_{\lambda_{j,max}} \end{pmatrix} \cdot \boldsymbol{S}^{(j)} \cdot \begin{pmatrix} \boldsymbol{r}^{(j)}_1(\boldsymbol{x}^{(j)}) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{r}^{(j)}_{\lambda_{j,max}}(\boldsymbol{x}^{(j)}) \end{pmatrix}. \end{aligned} \tag{15}$$

Clearly, the matrix $\boldsymbol{\Xi}^{(j)}(\boldsymbol{x}^{(j)})$ depends on the j-th component of the evaluation point $\boldsymbol{x}$, by the third factor. We recall that every $\boldsymbol{r}^{(j)}_i(\boldsymbol{x}^{(j)})$ is a vector with only a constant number of nonzero entries. So, in application, one has to decide whether to compute the matrix as in (15), by using the sparseness of the third matrix, for every evaluation point, or just compute the product of the first two dense matrices and keep the independence of the evaluation point.

Altogether, the tensor product multilevel interpolation or penalized least-squares approximation has the following form. This is the main result of this section and is the representation that allows the most pre-computation.

Theorem 3.8 *With the notation and assumptions made throughout, we can represent the tensor product multilevel approximation to* $\boldsymbol{f}$ *as*

$$\begin{aligned}\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(\boldsymbol{f})(\boldsymbol{x}) = \\ \sum_{\boldsymbol{\lambda}\in\mathcal{J}_{\boldsymbol{\omega}}(\ell,d)} \sum_{\substack{\boldsymbol{\beta}\in\{0,1\}^d \\ \boldsymbol{\lambda}+\boldsymbol{\beta}\in\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} (-1)^{\|\boldsymbol{\beta}\|_1+d} \\ \sum_{u_1^{(1)}\in\{1,\dots,\lambda_1\}} \cdots \sum_{u_1^{(d)}\in\{1,\dots,\lambda_d\}} \sum_{k_1^{(1)}\le N_{u_1^{(1)}}^{(1)}} \cdots \sum_{k_1^{(d)}\le N_{u_1^{(d)}}^{(d)}} \boldsymbol{f}(\boldsymbol{x}_{u_1,k_1})\cdot \\ \cdot \prod_{j=1}^{d} \sum_{m=u_1^{(j)}}^{\lambda_j} \left(\boldsymbol{\xi}_{u_1^{(j)},m}^{(j)}(\boldsymbol{x}^{(j)})\right)_{k_1^{(j)}}\end{aligned} \tag{16}$$

for every $\boldsymbol{x} \in \Omega^{(1)} \times \cdots \times \Omega^{(d)}$.

3.2 A Nodal Representation

In the previous section, we found a representation of the operator (7) by smartly finding ways to pre-compute certain terms. However, it still has several downsides. For one, we need the same value $\boldsymbol{f}(\boldsymbol{x}_{u_1,k_1})$ multiple times, which can be alleviated by storing all those values in an easy-to-access data structure. But more importantly, we have no advantage, if the direction-wise sites are nested, i.e., if

$$X_1^{(j)} \subset X_2^{(j)} \subset \cdots \subset X_{\lambda_{j,max}}^{(j)}.$$

It turns out that in this case, we can find another representation of the operator in (7). This new representation has the advantage that the outer sum is over the points in the associated sparse grid.

To do this, we need to introduce some more notation. In a nested family of sets of sites, for each point $\boldsymbol{x} \in X_{\lambda_{j,max}}^{(j)}$, there is a unique level $u(\boldsymbol{x}) \in \{1,\dots,\lambda_{j,max}\}$ where it first occurs. This point has position $k_j(\boldsymbol{x}, i_j)$ in the sets $X_{i_j}^{(j)}$, $u(\boldsymbol{x}) \le i_j \le \lambda_{j,max}$.

With this notation, we can find a new representation of the direction-wise multilevel operators.

Theorem 3.9 *With the notation and assumptions made throughout this paper, we define*

$$A_{\{u(\boldsymbol{x}),\dots,L\}}\chi_{L,k(\boldsymbol{x},L)} := \sum_{\emptyset\neq \mathrm{u}\subseteq\{u(\boldsymbol{x}),\dots,L\}} (-1)^{\#\mathrm{u}+1}\mathcal{I}_{\mathrm{u}}\chi_{L,k(\boldsymbol{x},L)}.$$

Then the multilevel operator introduced in Theorem 2.5 can be expressed as

$$A_L(f) := \sum_{\boldsymbol{y}\in X_L} f(\boldsymbol{y}) A_{\{u(\boldsymbol{y}),\dots,L\}}\chi_{L,k(\boldsymbol{y},L)}.$$

Using this representation of the direction-wise multilevel operators in the combination technique and easy manipulations of the sums yield the *nodal representation* of the tensor product multilevel operator.

Theorem 3.10 *With the notation and assumption made throughout this paper, we can represent the tensor product multilevel operator $\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}$, defined in Definition 2.6, as*

$$\begin{aligned}\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}(f) = &\sum_{\boldsymbol{x}\in\mathcal{H}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} f(\boldsymbol{x})\cdot \\ &\cdot \sum_{\substack{\boldsymbol{\lambda}\in\mathcal{J}_{\boldsymbol{\omega}}(\ell,d)\\ \boldsymbol{\lambda}\geq \boldsymbol{u}(\boldsymbol{x})}} \sum_{\substack{\boldsymbol{\beta}\in\{0,1\}^d\\ \boldsymbol{\lambda}+\boldsymbol{\beta}\in\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}} (-1)^{\|\boldsymbol{\beta}\|_1+d}\cdot \\ &\cdot \bigotimes_{j=1}^{d} A^{(j)}_{\{u_j(\boldsymbol{x}^{(j)}),\ldots,\lambda_j\}}\chi^{(j)}_{\lambda_j,k_j(\boldsymbol{x}^{(j)},\lambda_j)}.\end{aligned}$$

Remark 3.11 1. As a direct consequence, the nodal representation of the tensor product multilevel method in Theorem 3.10 is an ideal candidate for GPU parallelization: For every point in the sparse grid, we do more or less the same computations.

2. Again, we can pre-compute all the direction-wise multilevel approximations $A^{(j)}_{\{u_j(\boldsymbol{x}^{(j)}),\ldots,\lambda_j\}}$ $\chi^{(j)}_{\lambda_j,k_j(\boldsymbol{x}^{(j)},\lambda_j)}$ and keep them in memory. For a single evaluation of the approximation at a point $\boldsymbol{y} = (\boldsymbol{y}^{(1)},\ldots,\boldsymbol{y}^{(d)}) \in \Omega^{\otimes}$ we need to evaluate the stored multilevel approximations once and store them in an appropriate data structure such that getting the value $A^{(j)}_{\{u_j(\boldsymbol{x}^{(j)}),\ldots,\lambda_j\}}\chi^{(j)}_{\lambda_j,k_j(\boldsymbol{x}^{(j)},\lambda_j}(\boldsymbol{y}^{(j)})$ is a simple lookup.

4 Numerical Examples

We will now showcase two example applications for the tensor product multilevel method. In the first test case, the method is applied to interpolate simulation data from a shallow water simulation in space and time. There, the tensor product multilevel method is setup with a two-dimensional domain (the spatial domain of the simulation) in the first direction and a one-dimensional domain (the time) in the second direction.

The second example interpolates a seven-dimensional function on a sparse grid. The interpolated function in this case is the shear force of a cantilever beam, and the tensor product multilevel method is set up with completely isotropic directions.

4.1 Tidal Flow at the Bight of Abaco

In the first example, we apply the tensor product multilevel method to interpolate simulation data from a tidal flow simulation at the Bight of Abaco, a bay of the Bahamas. The tidal flow is simulated by the two-dimensional shallow water equations, given by

$$\begin{aligned}&\partial_t \xi + \nabla\cdot\boldsymbol{q} = 0,\\ &\partial_t \boldsymbol{q} + \nabla\cdot\left(\boldsymbol{q}\boldsymbol{q}^T/H\right) + \tau_{\text{bf}}\boldsymbol{q} + \begin{pmatrix}0 & -f_c\\ f_c & 0\end{pmatrix}\boldsymbol{q} + gH\nabla\xi = \boldsymbol{F},\end{aligned} \tag{17}$$

where ξ is the water height above some median sea level, $\boldsymbol{q} = (U, V)^T$ denotes the depth-integrated horizontal velocities, and $H = \xi + h_b$ is the total water depth relative to the bathymetry h_b. Other variables in the model are the Coriolis coefficient f_c, the bottom friction coefficient τ_{bf}, the gravitational acceleration g, and external forces $\boldsymbol{F}$.

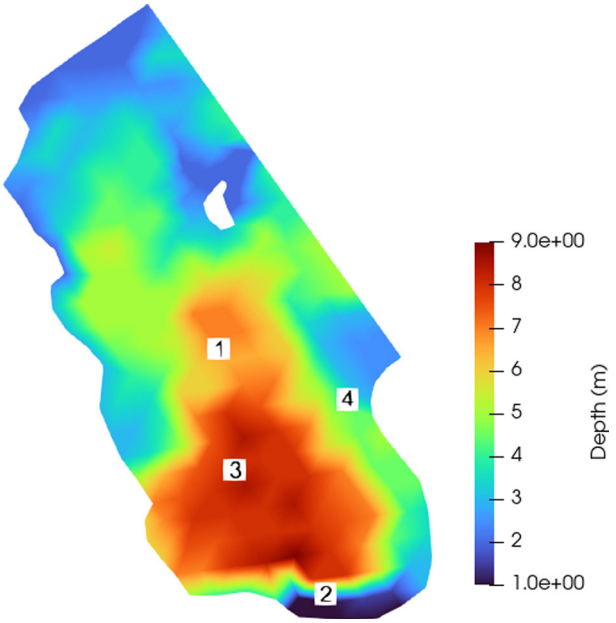


Fig. 1 Bathymetry for the Bight of Abaco. Horizontal extends of the domain are 70 km in x direction and 100 km in y direction. Four virtual recording stations report the variables ξ, U and V at the positions marked by 1 – 4

The PDE (17) is discretized with a discontinuous Galerkin method on unstructured triangular meshes. Details of the discretization and implementation can be found in [22]. Figure 1 shows the bathymetry of the Bight of Abaco, the exact boundary conditions and parameters for this domain can be found in [22, Section 2.3]. We run a tidal simulation with this bathymetry on a mesh with approximately 55,000 grid points for a simulation period of three days. After an initial ramp-up of two days for the boundary conditions, the variables ξ, U, and V are written for all grid points at intervals of five minutes during the third simulation day. This data then serves as input to the multilevel method.

The tensor product multilevel method is now set up with two directions. The first direction represents the two-dimensional spatial coordinate within the domain, and the second direction represents the time. We generate a set of nested points for the spatial direction by iteratively thinning the mesh points with an algorithm presented in [23], so that the separation distances q_i, i.e., the smallest distance of two sites, satisfy $q_{i+1} \approx \frac{1}{2} q_i$ for levels $i = 1, \ldots, 6$. On the coarsest level, this results in merely 35 points in the whole domain, whereas the finest level contains approximately 38,000 points. The resulting point sets are depicted in Fig. 2. We use the compactly supported kernel $\phi_{3,1}(r) = (1-r)_+^4(4r+1)$ for the RBF interpolation, and the support radius δ_i is coupled to the separation distance at each level by $\delta_i = 6q_i$. For the time direction, we used the initial output frequency of five minutes on the finest level and double the time step for each coarser level. As a basis function for interpolation, we use the kernel $\phi_{1,1}(r) = (1-r)_+^3(3r+1)$ with the support radius set to $\delta_i = 6\Delta t_i$. Table 1 summarizes the number of points, separation distance, and time step Δt for all levels.

We use the mesh with 55,000 points, which generated the original data, to evaluate the multilevel method using the representation in (16). By evaluating the interpolant on these points and comparing the result point-wise with the simulation data, we are able to quantify

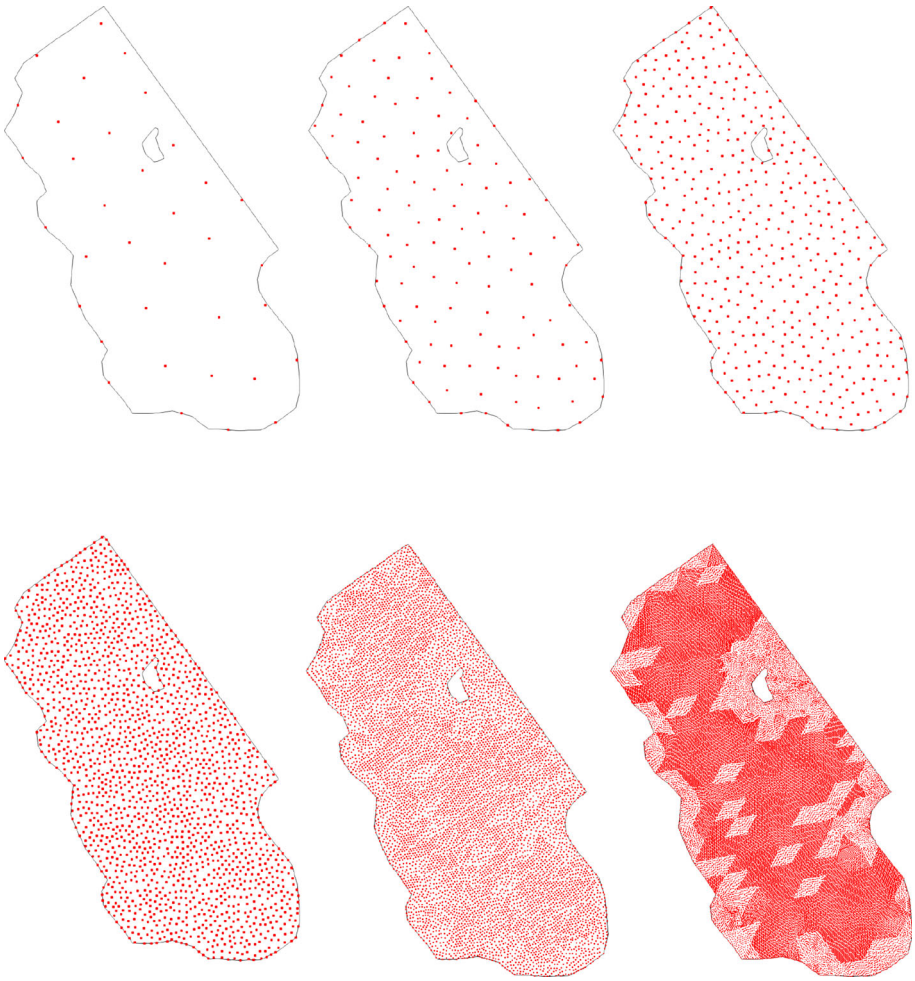


Fig. 2 Point sets used for the interpolation from coarsest to finest level. Because of the unstructured input mesh, the resulting point-sets of higher levels tend not to be quasi-uniform

Table 1 Number of points, separation distance and time step for each level for the Bight of Abaco simulation

Level	1	2	3	4	5	6
Number of points	35	123	476	1871	8228	37953
Separation distance (m)	9133.3	4521.2	2254.9	1128.1	563.7	281.9
Points in time direction	9	18	36	72	144	288
Δt (minutes)	160	80	40	20	10	5

the error of the multilevel method. Fig. 3 shows the difference between the interpolant and the simulation result for the variable U. The largest differences appear near the boundary of the domain, especially near the island. On the finest level the interpolation error is below 10^{-2} for 95 % of the mesh points.

Fig. 4 shows the average pointwise error for different number of levels and all three variables. The water height ξ is approximated better than the velocities and has a higher convergence order: For ξ, the computed order of convergence is approximately 2.0, whereas the convergence order of U and V is 1.6 and 1.5 respectively.

If we focus on a single recording station (Fig. 5 left), we see that the velocity profile from the interpolant matches the simulation data. However, there are also some slight deviations visible (Fig. 5 right). Because the station locations are within the elements, and not part of the interpolation sites, a slight deviation is not surprising.

4.2 Cantilever beam

As a second test example, we consider a cantilever beam, as described in [24] and depicted in Figure 6. The displacement field $\boldsymbol{u}$ satisfies the Navier-Lamé equation,

$$\begin{aligned}(\lambda(\boldsymbol{x},\boldsymbol{y})+\mu(\boldsymbol{x},\boldsymbol{y}))\nabla(\nabla\cdot\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}))+\mu(\boldsymbol{x},\boldsymbol{y})\nabla^2\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}) &= -\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) && \boldsymbol{x}\in\Omega\\ \sigma(\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}))\cdot\boldsymbol{n} &= 0 && \boldsymbol{x}\in\partial\Omega\setminus\partial\Omega_{wall}\\ \boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{0} && \boldsymbol{x}\in\partial\Omega_{wall},\end{aligned}$$

where μ and λ are the Lamé constants given by

$$\mu(\boldsymbol{x},\boldsymbol{y})=\frac{E(\boldsymbol{x},\boldsymbol{y})}{2(1+\nu)} \quad\text{and}\quad \lambda(\boldsymbol{x},\boldsymbol{y})=\frac{\nu E(\boldsymbol{x},\boldsymbol{y})}{(1+\nu)(1-2\nu)},$$

$\nu=0.28$ is Poisson's ratio, E is Young's modulus and σ is the Cauchy stress tensor computed by

$$\sigma(\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}))=\lambda\left(\nabla\cdot\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y})\right)I+\mu\left(\nabla\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y})+(\nabla\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}))^T\right).$$

As shown in the figure, the beam is separated into seven non-overlapping subdomains Ω_i, $i=1,\ldots,7$, each having a different value for Young's modulus:

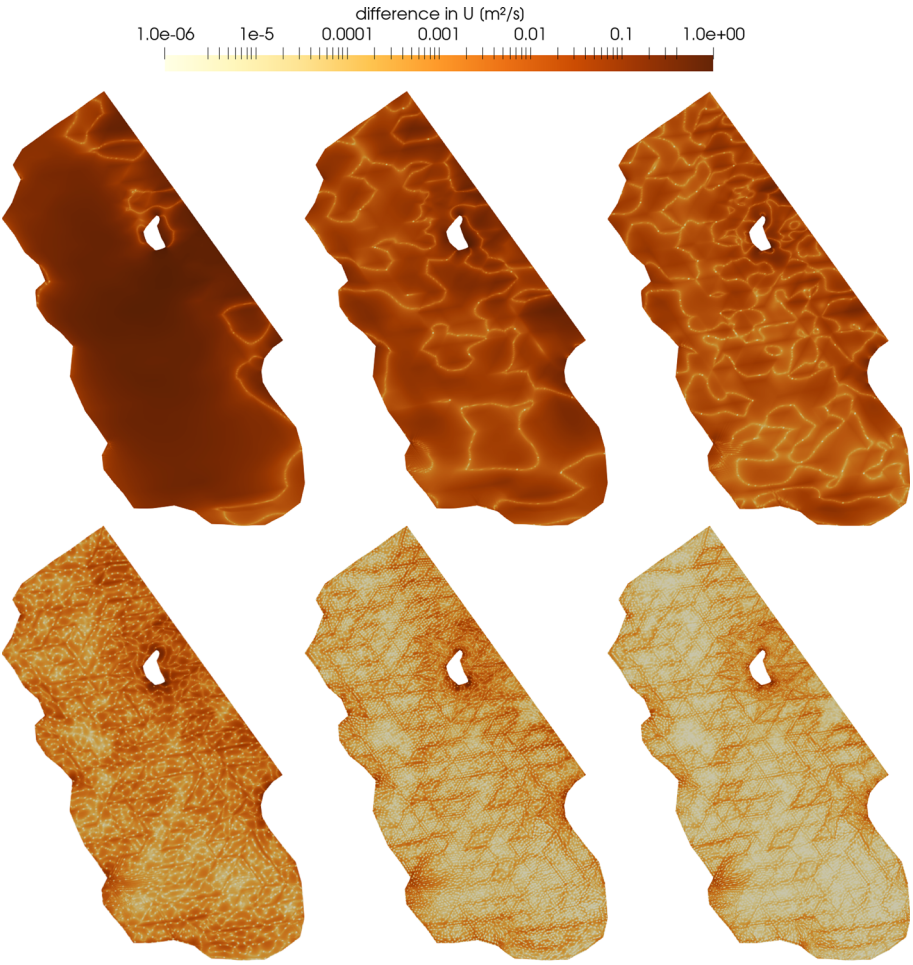
$$E(\boldsymbol{x},\boldsymbol{y})=\exp(7+y_i)\text{ if }\boldsymbol{x}\in\Omega_i,\ i=1,\ldots,7.$$

We set, following the description in [24], $\boldsymbol{f}\equiv 1$ and compute the total shear force in y-direction by

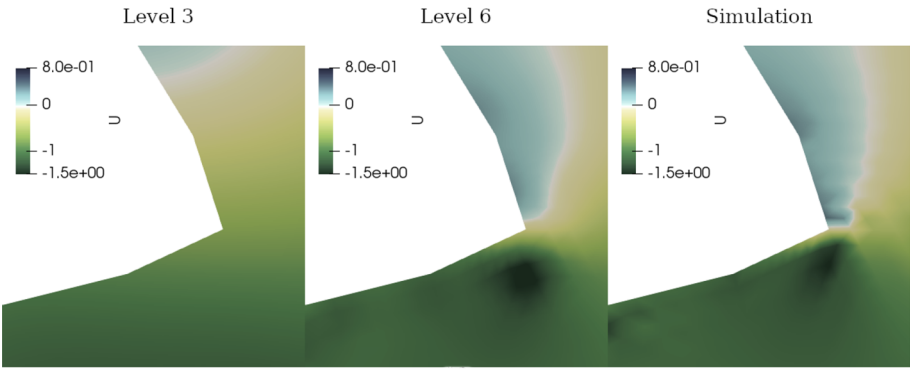
$$q(\boldsymbol{y})=\int_\Omega \sigma_{12}(\boldsymbol{u}(\boldsymbol{x},\boldsymbol{y}))\,d\boldsymbol{x}, \tag{18}$$

and use the tensor product multilevel method to approximate this function. We decompose the domain $[-1,1]^7$ into one-dimensional intervals, and generate a sequence of nested equidistant points in each direction, where level i contains 2^i+1 points. These points are then used in combination with the compactly supported basis functions

$$\begin{aligned}\phi_{1,1}(r)&=(1-r)_+^3(3r+1)\\ \phi_{1,2}(r)&=(1-r)_+^5(8r^2+5r+1)\end{aligned}$$



(a) Pointwise absolute error for levels 1 – 6.



(b) Difference in variable U for level 3, 6, and simulation "truth".

Fig. 3 Absolute pointwise difference for U on the whole mesh (top), simulation and interpolation result for U near the island (bottom)

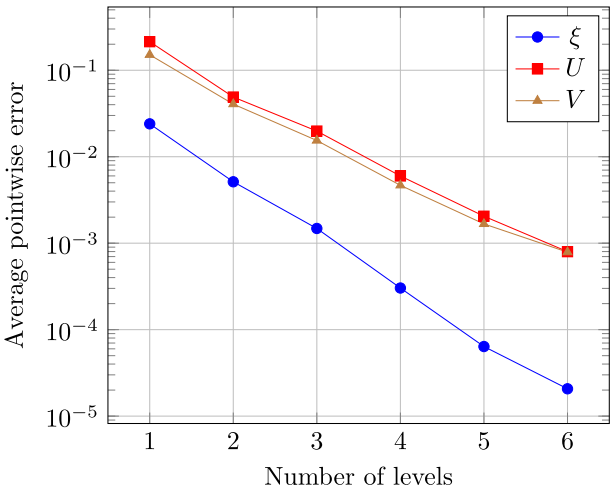


Fig. 4 Average pointwise error for the Bahamas example with different number of levels and variables

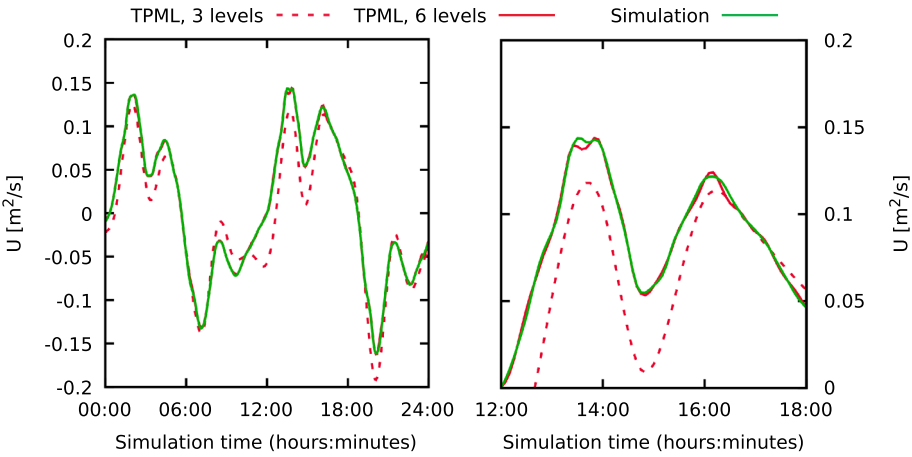


Fig. 5 Depth-integrated velocity in x direction at recording station 2

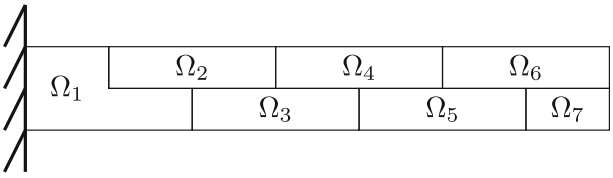


Fig. 6 Cantilever beam from [24, Section 3.4]

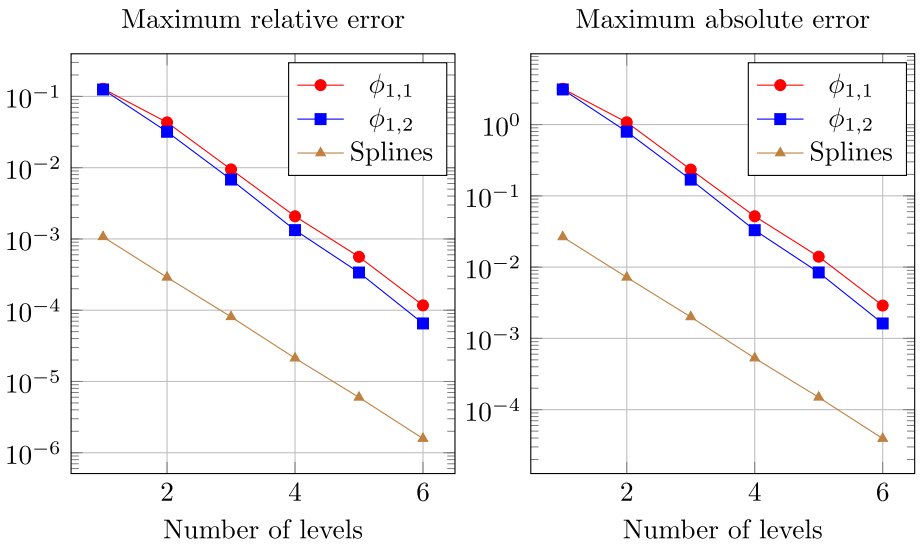


Fig. 7 Maximum relative and absolute error for the beam model with different number of levels. For the spline approximation, the "level" corresponds to the maximum value $\lambda_{j,max}$ for each direction in the index set

Table 2 Number of summands in the different representations: naive representation of (7), cardinal representation of (11) and the nodal representation in Theorem 3.10

Level	Naive	Cardinal	Nodal
1	315	315	315
2	47,097	2,052	1,422
3	20,300,247	9,454	6,021
4	$\approx 4 \cdot 10^{10}$	40,113	24,597
5	$\approx 3 \cdot 10^{14}$	171,117	106,407
6	$\approx 10^{19}$	757,863	480,339

for the tensor product multilevel method. The support radius is set to four times the separation distance. Figure 7 shows the maximum relative and absolute error for the beam model for both kernels.

4.3 Runtime and Aspects of Implementation

It is difficult to give general estimates for the complexity of the cardinal and nodal representations of the tensor product multilevel method. In the original representation (7) the inner sums for a fixed index $\boldsymbol{\lambda}$ iterate over subsets of $\{1, \dots, \lambda_i\}$ and then over combination of points on all levels, making the representation infeasible for applications. The cardinal representation from Theorem 3.8 reduces this complexity by only iterating over elements of $\{1, \dots, \lambda_i\}$ and points from one level. The contribution of the higher levels were moved into a matrix (cf. (15)), which can be partially computed in an offline-phase. Lastly, the nodal representation from Theorem 3.10 iterates over the points in the sparse grid and for each point then over a subset of index vectors $\boldsymbol{\lambda}$.

To get a feeling for the number of summands involved in each of the representations of $\mathcal{A}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}$ dicussed here, i.e., the naive representation of (7), cardinal representation of (11)

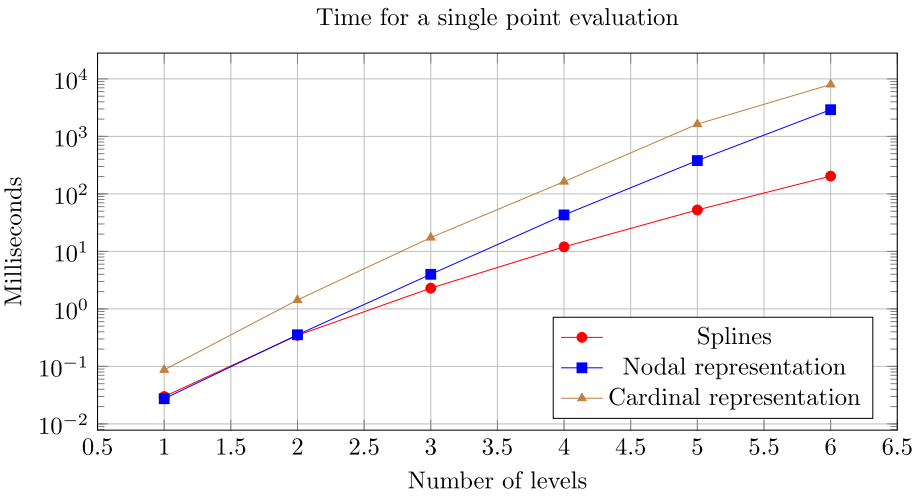


Fig. 8 Time for a single point evaluation for the Beam example in the previous subsection. The cardinal representation is the one of Section 3.1

and the nodal representation in Theorem 3.10, Table 2 shows the number of summands in all of these representations for the example in Section 4.1. Even though this neglects the costs of evaluating a single operand, it clearly shows that the original representation in (7) is practically infeasible, and the newly introduced cardinal and nodal representations show a much lower computational cost.

Fig. 8 shows the evaluation time for a single point in the beam example from the previous subsection. We compare the two new cardinal and nodal representations from this paper against a standard sparse grid method using linear splines. The runtime shown in Fig. 8 are averaged over 2000 points. Time required for initially setting up data structures is not included.

In the two examples from this paper we pre-computed the function values, since the target function $\boldsymbol{f}$ involves the solution of a PDE. The gathered data is therefore only available as a scattered set, and it is important to effectively find the interpolation points. This can be, for example, be achieved with a kd-tree. An alternative to tree-based data structures are associative maps, which can perform better than kd-trees. We can identify the evaluation points by their index (either the index in the sparse grid or the index in the respective directions) and then use this index as a key in an associative map.

In conclusion, we have presented two improvements on the naive representation of the approximation operator (7), a nodal and a cardinal representation. In the standard example of the cantilever beam, Section 4.2, both representations show a slightly worse but comparable behavior to splines. Here, additional reseach to further improve the methods is necessary. The TPML has the advantage to extend to product of low dimensional domains, where the classical spline based method is not applicable.

A Proofs of the statements of Section 3.1

Proof of Lemma 3.1 We recall the representation of $\mathcal{I}_{\mathfrak{u}}$ in (6),

$$\mathcal{I}_{\mathfrak{u}} = \sum_{\boldsymbol{k} \le \boldsymbol{N}_{\mathfrak{u}}} f(\boldsymbol{x}_{u_1,k_1}) a(\mathfrak{u}, \boldsymbol{k}) \chi_{\mathfrak{u}_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}.$$

Inserting the representations for the Lagrange functions (4), yields

$$\begin{aligned}\mathcal{I}_{\mathfrak{u}}(f)(\boldsymbol{x}) &= \\ &= \sum_{\boldsymbol{k} \le \boldsymbol{N}_{\mathfrak{u}}} f(\boldsymbol{x}_{u_1,k_1}) \left[\prod_{m=1}^{\#\mathfrak{u}-1} \boldsymbol{\alpha}_{u_m,k_m}^{\mathrm{T}} \boldsymbol{r}_{u_m}(\boldsymbol{x}_{u_{m+1},k_{m+1}}) \right] \boldsymbol{\alpha}_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}^{\mathrm{T}} \boldsymbol{r}_{u_{\#\mathfrak{u}}}(\boldsymbol{x}).\end{aligned}$$

We see that in the product in the brackets, the index u_1, k_1 only appears in $\boldsymbol{\alpha}_{u_1,k_1}$. Similarly, in the product, there is no $\boldsymbol{\alpha}_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}^{\mathrm{T}}$. This vector is written explicitly after the bracket. Hence, we can split $\boldsymbol{\alpha}_{u_1,k_1}^{\mathrm{T}}$ from the product and add $\boldsymbol{\alpha}_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}^{\mathrm{T}}$ to it. This yields, after an index shift in the product term,

$$\begin{aligned}\mathcal{I}_{\mathfrak{u}}(f)(\boldsymbol{x}) &= \\ &= \sum_{\boldsymbol{k} \le \boldsymbol{N}_{\mathfrak{u}}} f(\boldsymbol{x}_{u_1,k_1}) \boldsymbol{\alpha}_{u_1,k_1}^{\mathrm{T}} \left[\prod_{m=2}^{\#\mathfrak{u}} \boldsymbol{r}_{u_{m-1}}(\boldsymbol{x}_{u_m,k_m}) \boldsymbol{\alpha}_{u_m,k_m}^{\mathrm{T}} \right] \boldsymbol{r}_{u_{\#\mathfrak{u}}}(\boldsymbol{x}).\end{aligned}$$

Next, we split the multiple sums $\sum_{\boldsymbol{k} \le \boldsymbol{N}_{\mathfrak{u}}}$ in its single components. Only the terms $f(\boldsymbol{x}_{u_1,k_1})$ and $\boldsymbol{\alpha}_{u_1,k_1}^{\mathrm{T}}$ depend on k_1. We pull the other sums into the product. We have

$$\begin{aligned}\mathcal{I}_{\mathfrak{u}}(f)(\boldsymbol{x}) = \sum_{k_1 \le N_1} f(\boldsymbol{x}_{u_1,k_1}) \boldsymbol{\alpha}_{u_1,k_1}^{\mathrm{T}} \cdot \\ \cdot \left[\prod_{m=2}^{\#\mathfrak{u}} \sum_{k_m \le N_m} \boldsymbol{r}_{u_{m-1}}(\boldsymbol{x}_{u_m,k_m}) \boldsymbol{\alpha}_{u_m,k_m}^{\mathrm{T}} \right] \boldsymbol{r}_{u_{\#\mathfrak{u}}}(\boldsymbol{x}).\end{aligned}$$

This is the claim if we define $P_{\mathfrak{u}}$ as in (9). □

Proof of Theorem 3.3 The statement in Theorem 3.3 follows easily from the observation that

$$\left(\chi_{i_1}^{(1)} \otimes \cdots \otimes \chi_{i_d}^{(d)} \right)(\boldsymbol{x}) = \prod_{j=1}^{d} \chi_{i_j}^{(j)}(\boldsymbol{x}^{(j)})$$

and then applying Lemma 3.1 in every direction independently. □

Proof of Corollary 3.4 As already outlined in the text, this is an application of Theorem 3.3, together with using the observation that we only use evaluations of $\boldsymbol{f}$ in points on level $u_1^{(j)}$. The claim is then obtained by clever grouping of the different summands. □

The proofs of the remaining Corollary 3.6 and Theorem 3.8 were already outlined in the text.

B Proofs of the statements of Section 3.2

Proof of Theorem 3.9 Following Theorem 2.5, the multilevel operator A_L has the representation

$$A_L(f) = \sum_{\substack{\mathfrak{u}\subseteq\{1,\dots,L\}\\ 1\le\#\mathfrak{u}\le L}} (-1)^{\#\mathfrak{u}+1} \sum_{\boldsymbol{k}\le \boldsymbol{N}_\mathfrak{u}} a(\mathfrak{u},\boldsymbol{k}) f(\boldsymbol{x}_{u_1,k_1}) \chi_{u_{\#\mathfrak{u}},k_{\#\mathfrak{u}}}.$$

Again, we split off the first elements of the ordered sets $\mathfrak{u}$ and $\boldsymbol{k}$, yielding the decompositions $\mathfrak{u} = \{u_1\}\cup\widetilde{\mathfrak{u}}$ and $\boldsymbol{k} = (k_1, \widetilde{\boldsymbol{k}})$. Here, $\widetilde{\mathfrak{u}} \subseteq \{u_1+1,\dots,L\}$ and $\widetilde{\boldsymbol{k}} \le \boldsymbol{N}_{\widetilde{\mathfrak{u}}}$ and $\widetilde{\mathfrak{u}} = \emptyset$ is allowed and means that $\mathfrak{u} = \{u_1\}$ and $\boldsymbol{k} = (k_1)$. This allows us to write

$$A_L(f) = \sum_{u_1\in\{1,\dots,L\}} \sum_{k_1\le N_{u_1}} f(\boldsymbol{x}_{u_1,k_1}) \sum_{\substack{\widetilde{\mathfrak{u}}\subseteq\{u_1+1,\dots,L\}\\ \mathfrak{u}=\{u_1\}\cup\widetilde{\mathfrak{u}}}} (-1)^{\#\mathfrak{u}+1} \cdot \sum_{\widetilde{\boldsymbol{k}}\le\boldsymbol{N}_{\widetilde{\mathfrak{u}}}} \left(\prod_{\ell=1}^{\#\mathfrak{u}-1} \chi_{u_\ell,k_\ell}(\boldsymbol{x}_{u_{\ell+1},k_{\ell+1}})\right) \chi_{u_{\#\mathfrak{u}},k_{\#\boldsymbol{k}}}.$$

Now we want to rewrite this expression in terms of the points X_L on level L. For a fixed point $x^* \in X_L$, there is a unique level $u(\boldsymbol{x}^*)$ on which the point occurs first in $X_{u(\boldsymbol{x}^*)}$. By grouping the terms by the points $\boldsymbol{x} \in X_L$ in the representation above, we arrive at

$$A_L(f) = \sum_{\boldsymbol{x}\in X_L} f(\boldsymbol{x}) \sum_{u_1\in\{u(\boldsymbol{x}),\dots,L\}} \sum_{\substack{\widetilde{\mathfrak{u}}\subseteq\{u_1+1,\dots,L\}\\ \mathfrak{u}=\{u_1\}\cup\widetilde{\mathfrak{u}}}} (-1)^{\#\mathfrak{u}+1} \cdot \sum_{\substack{\boldsymbol{k}\le\boldsymbol{N}_\mathfrak{u}\\ k_1=\bar{k}(\boldsymbol{x},u_1)}} \left(\prod_{\ell=1}^{\#\mathfrak{u}-1} \chi_{u_\ell,k_\ell}(\boldsymbol{x}_{u_{\ell+1},k_{\ell+1}})\right) \chi_{u_{\#\mathfrak{u}},k_{\#\boldsymbol{k}}}.$$

Next, we realize that the term

$$\sum_{\substack{\boldsymbol{k}\le\boldsymbol{N}_\mathfrak{u}\\ k_1=\bar{k}(\boldsymbol{x},u_1)}} \left(\prod_{\ell=1}^{\#\mathfrak{u}-1} \chi_{u_\ell,k_\ell}(\boldsymbol{x}_{u_{\ell+1},k_{\ell+1}})\right) \chi_{u_{\#\mathfrak{u}},k_{\#\boldsymbol{k}}}$$

is the expression $\mathcal{I}_{\mathfrak{u}}\chi_{L,k(\boldsymbol{x},L)}$, i.e., the combined operator of (6) applied to the Lagrange function on level L centered in the point with index $k(\boldsymbol{x}, L)$ in X_L, we obtain

$$\begin{aligned}
&\sum_{u_1\in\{u(\boldsymbol{x}),\dots,L\}} \sum_{\substack{\widetilde{\mathfrak{u}}\subseteq\{u_1+1,\dots,L\}\\ \mathfrak{u}=\{u_1\}\cup\widetilde{\mathfrak{u}}}} (-1)^{\#\mathfrak{u}+1}\cdot\\
&\qquad\cdot \sum_{\substack{\boldsymbol{k}\le \boldsymbol{N}_{\mathfrak{u}}\\ k_1=k(\boldsymbol{x},u_1)}} \left(\prod_{\ell=1}^{\#\mathfrak{u}-1}\chi_{u_\ell,k_\ell}(\boldsymbol{x}_{u_{\ell+1},k_{\ell+1}})\right)\chi_{u_{\#\mathfrak{u}},k_{\#\boldsymbol{k}}}\\
&= \sum_{u_1\in\{u(\boldsymbol{x}),\dots,L\}} \sum_{\substack{\widetilde{\mathfrak{u}}\subseteq\{u_1+1,\dots,L\}\\ \mathfrak{u}=\{u_1\}\cup\widetilde{\mathfrak{u}}}} (-1)^{\#\mathfrak{u}+1}\mathcal{I}_{\mathfrak{u}}\chi_{L,k(\boldsymbol{x},L)}\\
&= \sum_{\emptyset\neq\mathfrak{u}\subseteq\{u(\boldsymbol{x}),\dots,L\}} (-1)^{\#\mathfrak{u}+1}\mathcal{I}_{\mathfrak{u}}\chi_{L,k(\boldsymbol{x},L)}\\
&=: A_{\{u(\boldsymbol{x}),\dots,L\}}\chi_{L,k(\boldsymbol{x},L)}.
\end{aligned}$$

Hence, we have the representation

$$A_L(f) = \sum_{\boldsymbol{y}\in X_L} f(\boldsymbol{y})A_{\{u(\boldsymbol{y}),\dots,L\}}\chi_{L,k(\boldsymbol{y},L)},$$

that was claimed in Theorem 3.9. □

Proof of Theorem 3.10 The claim of Theorem 3.10 follows from inserting the representation of the direction-wise multilevel operator derived in Theorem 3.9 into the combination technique representation of the Smolyak operator, together with the observation, that for every component $\boldsymbol{x}^{(j)}$ of elements of the sparse grid $\mathcal{H}_{\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)}$ occurs first on level $u_j(\boldsymbol{x}^{(j)})$ and hence, is contained in $X^{(1)}_{\lambda_1}\times\cdots\times X^{(d)}_{\lambda_d}$ for $\boldsymbol{\lambda}\in\mathcal{I}_{\boldsymbol{\omega}}(\ell,d)$ if and only if $\lambda_j \ge u_j(\boldsymbol{x}^{(j)})$, $1\le j\le d$. This then yields the claimed nodal representation. □

Acknowledgements We thank Vadym Aizinger for kindly providing the data used in Section 4.1.

Funding Open Access funding enabled and organized by Projekt DEAL. The work of Holger Wendland was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 452806809. Funded by the Open Access Publishing Fund of the University of Bayreuth.

Data Availability No openly accessible data were used in the presented research. The used data was produced at the chair of Scientific Computing at the University of Bayreuth by simulation.

Declarations

Competing interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Smoljak, S.A.: Quadrature and interpolation formulae on tensor products of certain function classes. Dokl. Akad. Nauk SSSR **148**, 1042–1045 (1963)
2. Gerstner, T., Griebel, M.: Numerical integration using sparse grids. Numer. Algorithms **18**(3), 209–232 (1998)
3. Gerstner, T., Griebel, M.: Dimension-adaptive tensor-product quadrature. Computing **71**(1), 65–87 (2003). https://doi.org/10.1007/s00607-003-0015-5
4. Haji-Ali, A.-L., Harbrecht, H., Peters, M.D., Siebenmorgen, M.: Novel results for the anisotropic sparse grid quadrature. J. Complexity **47**, 62–85 (2018)
5. Conrad, P.R., Marzouk, Y.M.: Adaptive smolyak pseudospectral approximations. SIAM J. Sci. Comput. **35**(6), 2643–2670 (2013)
6. Barthelmann, V., Novak, E., Ritter, K.: High dimensional polynomial interpolation on sparse grids. Adv. Comput. Math. **12**, 273–288 (2000)
7. Kempf, R., Wendland, H.: High-dimensional approximation with kernel-based multilevel methods on sparse grids. Numer. Math. **154**(3–4), 485–519 (2023). https://doi.org/10.1007/s00211-023-01363-x
8. Bungartz, H.-J., Griebel, M.: A note on the complexity of solving poisson's equation for spaces of bounded mixed derivatives. J. Complexity **15**(2), 167–199 (1999). https://doi.org/10.1006/jcom.1999.0499
9. Nobile, F., Tempone, R., Webster, C.G.: An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. SIAM J. Numer. Anal. **46**(5), 2411–2442 (2008)
10. Griebel, M., Hamaekers, J.: Sparse grids for the schrödinger equation. ESAIM: Math. Model. Numer. Anal. **41**(2), 215–247 (2007)
11. Garcke, J.: A dimension adaptive sparse grid combination technique for machine learning. In: Read, W., Larson, J.W., Roberts, A.J. (eds.) Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006. ANZIAM J., vol. 48, pp. 725–740 (2007)
12. Yang, J., Li, G.: On Sparse Grid Interpolation for American Option Pricing with Multiple Underlying Assets. arXiv:2309.08287 (2023)
13. Garcke, J.: Sparse grids in a nutshell. In: Garcke, J., Griebel, M. (eds.) Sparse Grids and Applications, pp. 57–80. Springer, Berlin, Heidelberg (2013)
14. Bungartz, H.-J., Griebel, M.: Sparse grids. Acta Numer. **13**, 147–269 (2004). https://doi.org/10.1017/S0962492904000182
15. Wendland, H.: Multiscale analysis in sobolev spaces on bounded domains. Numer. Math. **116**(3), 493–517 (2010)
16. Floater, M.S., Iske, A.: Multistep scattered data interpolation using compactly supported radial basis functions. J. Comput. Appl. Math. **73**(1), 65–78 (1996)
17. Wendland, H.: Multiscale radial basis functions. In: Frames and Other Bases in Abstract and Function Spaces: Novel Methods in Harmonic Analysis. Volume 1, pp. 265–299. Birkhäuser, Cham (2017). https://eref.uni-bayreuth.de/55241/
18. Le Gia, Q.T., Sloan, I.H., Wendland, H.: Multiscale analysis in sobolev spaces on the sphere. SIAM J. Numer. Anal. **48**(6), 2065–2090 (2010)
19. Georgoulis, E.H., Levesley, J., Subhan, F.: Multilevel sparse kernel-based interpolation. SIAM J. Sci. Comput. **35**(2), 815–831 (2013). https://doi.org/10.1137/110859610
20. Rieger, C., Wendland, H.: Sampling inequalities for anisotropic tensor product grids. IMA J. Numer. Anal. **40**(1), 285–321 (2019)
21. Hackbusch, W.: Tensor Spaces and Numerical Tensor Calculus. Springer Series in Computational Mathematics. Springer, Berlin Heidelberg (2012). https://books.google.de/books?id=a5P71o6xcNMC
22. Büttner, M., Alt, C., Kenter, T., Köstler, H., Plessl, C., Aizinger, V.: Enabling Performance Portability for Shallow Water Equations on CPUs, GPUs, and FPGAs with SYCL. In: Proceedings of the Platform for Advanced Scientific Computing Conference, pp. 1–12. ACM, Zurich, Switzerland (2024). https://doi.org/10.1145/3659914.3659925
23. Schlömer, T., Heck, D., Deussen, O.: Farthest-point optimized point sets with maximized minimum distance. In: Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, pp.

135–142. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2018323.2018345
24. Migliorati, G., Nobile, F., Schwerin, E., Tempone, R.: Approximation of quantities of interest in stochastic pdes by the random discrete L^2 projection on polynomial spaces. SIAM J. Sci. Comput. **35**(3), 1440–1460 (2013)